

# DYNAMIC PETRI-NETS: A NEW MODELING TECHNIQUE FOR THE TOPOLOGY OF DISTRIBUTED SENSOR NETWORKS

Dr. Charles J. Graff  
US Army CERDEC  
Ft. Monmouth NJ 07703

## ABSTRACT

This paper introduces and describes a new extension to Petri-Nets that provide additional mechanisms for the modeling of dynamic, distributed and concurrent behavior. The extensions provide for the creation and destruction of conventional Petri-Net Places and Transitions providing dynamic behavior of Petri-Net operation. We call this new Petri-Net model Dynamic Petri-Nets (DPN). With this new model, the structure of the Petri-Net, i.e. the interconnection of Places and Transitions, will evolve over time. The paper introduces specific rules for the Place/Transition modifications, and presents some new and well known properties of Petri-Nets under these modifications. A Many Sorted Algebraic model of the DPN is presented that formally describes the techniques and relationships presented informally. Finally, an example of the use of these extensions is presented to represent the broadcast nature of mobile ad hoc and distributed sensor networks.

## 1. INTRODUCTION

In mobile ad hoc networks and sensor networks, it is critical to accurately represent dynamic behavior in a precise and rigorous fashion. The dynamic behavior includes not only state change, but also changes in structure. A number of mathematical techniques have been developed over the years to address this problem, including finite state machines and Petri-Nets, but they have focused primarily on the representation of state change.

Finite state machines have addressed the problem of representation dynamic behavior only through a change in state. The mathematical model of a finite state machine includes a finite set of states and a set of arcs interconnecting those states, a set of inputs, a set of outputs, and a unique start state and a set of end of terminating states. While both deterministic and non-deterministic state machines have been studied, it is well known that every non-deterministic state machine has an equivalent deterministic finite state machine, and hence provides no further increase in dynamic representation. The change in state is driven by some input stimulus

which may cause a state change and generate some output.

Petri-Nets were first created by Carl Petri in his dissertation "Kommunikation mit Automaten" in 1962. Petri-Nets are essentially bi-partite directed graphs, where the nodes are called either a Place or a Transition. A Place is similar to a state in a classic finite state machine, while a Transition is a representation of some type of action or activity. Entities called Tokens, which are typically integers, are assigned to Places, and, "move" through the Petri-Net from Place to Place under the action of Transition firing or executing. A set of specific rules govern the firing or execution of a Transition, and hence the movement of Tokens from Place to Place. Petri-Nets have the ability to represent synchronization, concurrency and non-determinism. While finite state machines and conventional Petri-Nets represent the concept associated with a change of state, they are very limited in their ability to represent changes in structure, and to represent those structural changes as an explicit function of time.

## 2. PROBLEM STATEMENT

The main technical challenge is to find a way to represent the structural changes of network behavior, consistent with the conventional representation of dynamic state change, synchronization, concurrency, non-determinism. The representation must be mathematically based and explicitly represent timed structural behavior.

This research creates and develops a new extension to Petri-Nets that provide additional mechanisms for the modeling of dynamic distributed and concurrent behavior. We call this new type of Petri Net a Dynamic Petri-Net (or DPN). The extensions provide for the creation and destruction of both conventional Petri-Net Places and Transitions through the dynamic behavior of Petri-Net operation. Thus the structure of the Petri-Net, i.e. the interconnection of Places and Transitions, may evolve over time. Thus the DPN can represent both structural change, as well as conventional state change (as represented by different Markings). It is anticipated that this work will support ongoing research efforts in

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>01 NOV 2006</b>		2. REPORT TYPE <b>N/A</b>		3. DATES COVERED <b>-</b>	
4. TITLE AND SUBTITLE <b>Dynamic Petri-Nets: A New Modeling Technique For The Topology Of Distributed Sensor Networks</b>				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>US Army CERDEC Ft. Monmouth NJ 07703</b>				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release, distribution unlimited</b>					
13. SUPPLEMENTARY NOTES <b>See also ADM002075., The original document contains color images.</b>					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT <b>UU</b>	18. NUMBER OF PAGES <b>8</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

Network Science (National Research Council 2005). In order to add the proper context, we first review conventional Petri-Nets.

### 3. REVIEW OF CONVENTIONAL PETRI-NETS

A conventional Petri-Net is represented as a 4-tuple:

$$PN = (P, T, A, M)$$

where

$P$  is a finite set of distinct Places, where each Place may contain zero or more integer Tokens,

$T$  is a finite set of distinct Transitions, (which are in effect Boolean functions)

$A$  is a set of directed and optionally weighted edges, whose members are represented by the ordered pair  $(p, t)$  or  $(t, p)$  (where the edge may have a tuple of attributes) and  $p$  is a member of the set  $P$ , and  $t$  is a member of the set  $T$ .

$M$  is the initial Marking of the Petri-Net (the initial assignment of Tokens to Places). We assume that the Tokens are indistinguishable, and are used to mark the Petri-Net. While they are typically integer values, they may also be any type of object.

We note that some Places may have only incoming (outgoing) arcs corresponding to final (initial) Places. Similarly, we note that some Transitions may have only incoming (outgoing) arcs as well.

Formally,

$$PN = (P, T, A, M)$$

where

$$P = \{p_1, p_2, p_3, \dots, p_n\} \quad \text{fixed } n \geq 1$$

$$T = \{t_1, t_2, t_3, \dots, t_m\} \quad \text{fixed } m \geq 1$$

$$P \cap T = \emptyset$$

$$A \subseteq (P \times T) \cup (T \times P)$$

$$M(t) = (m_1(t), m_2(t), m_3(t), \dots, m_n(t)) \text{ the initial marking}$$

We also define the reachability graph (may be finite or infinite) as a directed graph whose nodes are the individual and distinct Markings, and the edges are the Transitions that inter-connect the Markings. Thus

$$RG = (M, T)$$

where

$M$  is the (possibly infinite) set of Markings,

$T$  is the set of Transitions interconnecting the Markings  $M$ .

Further details about conventional Petri-Nets may be found in (Murata, T., 1989; Zuraski, R., and Zhou, M. 1994; and Sreenivas, R.S., 2006).

### 4. THE DYNAMIC PETRI-NET (DPN) MODEL

In conventional PN, the sets of  $P$ ,  $T$ , and  $A$  are FIXED, and the dynamic behavior is represented ONLY through the set of Markings, as Tokens move from Place to Place. The DPN provides the capability of Places, Transitions, or both to change as a function of time. These additional behaviors are as indicated below.

#### 4.1 A Conceptual View of the DPN

As indicated above, the essential idea for DPN is that we will permit the PN structure (that is the sets  $P$ ,  $T$ , and  $A$ ) to evolve over time by the addition or deletion of Places and/or Transitions. In general we term the structure (or bi-partite graph) of a DPN as a **configuration**. We model the action of Place or Transition insertion (or removal) into a sequence of atomic actions as follows:

1. The creation (or deletion) of a Place or Transition to/from DPN.
2. The creation, deletion, or modification of the connections between Places and Transitions.
3. The creation, deletion, or modification of the connections between Transitions and Places.

We note that in this formulation, it is possible to have a system that does not completely conform to the conventional notation of a PN, i.e. we admit that there may be isolated Places or Transitions that are not connected to the rest of the DPN. We term these as a **free-Place** or a **free-Transition**. We note that atomic action 1 must be completed before atomic actions 2 through 3 can take place.

While the above actions represent changes in the DPN structure, we still must address the issue of what to do with the Tokens associated with a Place that is affected by a change. We note that there are several options:

1. If the Place is removed, then either the Tokens contained within that Place are lost permanently, or
2. The Tokens are retained under the condition that that Place may be “reborn” or re-inserted

into the DPN with the same number of Tokens as when it was removed.

In order to maintain complete flexibility, we admit either option in our model for DPN.

If either a Place or Transition is removed, then all arcs incident or exiting that object are also removed. We require this to prevent “dangling” arcs connecting the Places and Transitions. In addition, we assume that this action takes place recursively as well, so that the removal of a Place may cause one or more Transitions to be removed as well.

If a Place is created, then either the Place is created containing zero Tokens, or we may admit that a newly created Place may be given some initial set of Tokens, corresponding to an initial state. Again, in order to maintain complete flexibility, we admit either option in our model for DPN.

A classic PN does not include any specific reference to time and are inherently asynchronous in nature. However, we note that some PN extensions (a Timed PN) may include explicit time delays on Transition firing (after being enabled) or explicit time delay on Token insertion into Places (i.e. a Time PN). In this fashion, the movement of Tokens and the related Marking vector changes are in fact time dependent. However in both cases the structure of the PN, i.e. Places, Transitions, and Arcs are time invariant.

In a typical operation, it is assumed that one or more Transitions may execute while the DPN is in configuration  $DPN_1$  and before the configuration change has taken place. Therefore we need a representation regarding the DPN behavior before and after the configuration change. We do this through the use of the Marking vector (i.e. a vector that indicates the integer number of Tokens residing in each Place of the PN) as described below.

#### 4.2 The Universal Set for Places (or Transitions) and Universal DPN

In order to simplify the DPN model we assume that there is a bounded “Universal Set” from which all the Places or Transitions are to be taken for the DPN. What this essentially means is that we have an upper bound on the total number of Places,  $P_{max}$ , and the total number of Transitions,  $T_{max}$ , that are possible in the DPN. If any Place  $P_i$  (or Transition  $T_j$ ) is associated with the DPN structure, then it is termed an **active**, **alive** or **enabled** Place (Or Transition); otherwise that Place (or Transition) is termed to be **inactive**, **dead** or **disabled**. As noted above, we do permit the Places or Transitions

to “**change state**”, by becoming active (alive) or inactive (dead) or vice versa.

Furthermore, we define an active Place or Transition as being **connected** (meaning having at least one input or output) or **isolated**, implying that the Place or Transition, while still in the DPN, is not connected to the rest of the DPN structure. Clearly any isolated Place or Transition can have no role in the firing or execution (the movement of Tokens) in a DPN.

#### 4.3 A Functional Representation of DPN

We let DPN represent a Dynamic Petri-Net

$$DPN = (P(t), T(t), A(t), M(t))$$

Where

$P(t)$  = set of Places where the number of Places is a function of time  $t$ ,

$T(t)$  = set of Transitions where the number of Transitions is a function of time,

$A(t)$  = set of Arcs which is a function of time

$M(t)$  = set of Markings which is a function of time.

More formally,

$$DPN = (P(t), T(t), A(t), M(t))$$

$$P(t) = \{p_1, p_2, p_3, \dots, p_n\}; n = n(t) \leq P_{max}$$

$$T(t) = \{t_1, t_2, t_3, \dots, t_m\}; m = n(t) \leq T_{max}$$

$$P(t) \cap T(t) = \phi$$

$$A(t) \subseteq (P(t) \times T(t)) \cup (T(t) \times P(t))$$

$$M(t) = (m_1(t), m_2(t), m_3(t), \dots, m_n(t)); n = n(0) = P_{max}$$

$$m_i \in \mathbb{Z}^+ \cup \{0\}$$

We represent the **DPN configuration change** as follows:

$$DPN_1(P, T, A, M) \rightarrow DPN_2(P', T', A', M')$$

where  $DPN_1$  is the PN before the configuration change has taken place, and  $DPN_2$  is the PN after the configuration change has taken place.

We note that  $DPN_1$  and  $DPN_2$  may have a different number of Places, Transitions, or interconnections as indicated above.

In a typical operation, it is assumed that one or more enabled Transition may **execute** or **fire** while the PN is in either configuration  $DPN_1$  or configuration  $DPN_2$ .

#### 4.4 A Many Sorted Algebra for DPN

In this section a more rigorous setting shall be established for the DPN using a Many Sorted Algebra. The Many Sorted Algebra will allow us to precisely specify as a function of time the behavior of the DPN including configuration changes.

The Many Sorted Algebraic Model of a DPN is presented below. We use the standard Many Sorted Algebraic notation as described in Giardina, C., 2000. More details regarding the definition of a Many Sorted Algebra is provided in the above reference.

In particular, we require that the Many Sorted Algebra representation of DPN include functions involving elements from Carrier Sets of Sorts, and constraints involving these functions and elements of Sorts. Figure 1 provides a pictorial representation of the DPN.

The Sorts are: Time (Ti), Transitions (T), Places (P), Tokens (To), Values (V). The actual elements utilized of the sorts involved above are called Carrier Sets, and are denoted by:  $A_{Ti}$ ,  $A_T$ ,  $A_P$ ,  $A_{To}$ ,  $A_V$  respectively. All of these sets are nonempty. We also assume, as

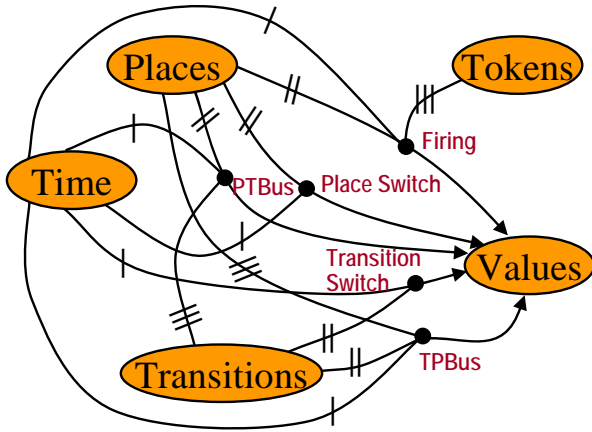


Figure 1. The Many Sorted Algebraic Model for a DPN System.

usual, the set  $A_V$  is Boolean. The Functions of the Many Sorted Algebra are:

1. Transition Switch – a function of time that enables or disables Transitions and is denoted by  $g$ . Hence:

$$g: A_{Ti} \times A_T \rightarrow A_V$$

where  $g(t,n) = 1$  means that at time  $t$  Transition  $n$  is alive and  $g(t,n) = 0$  means that at time  $t$  Transition  $n$  is dead.

2. Place Switch – a function of time that enables or disables Places is denoted by  $f$ . Hence:

$$f: A_{Ti} \times A_P \rightarrow A_V$$

where  $f(t,x) = 1$  means that at time  $t$  Place  $x$  is alive and  $f(t,x) = 0$  means that at time  $t$  Place  $x$  is dead.

3. PTBus – a function that provides a Token path from a fixed Place to several Transitions and is denoted by  $F$ . Hence:

$$F: A_{Ti} \times A_P \times A_T \rightarrow A_V$$

where  $F(t,x,n) = 1$  means there is a bus path from Place  $x$  to Transition  $n$  at time  $t$ . (A constraining equation will allow this to happen only when **both**  $g(t,n) = 1$  and  $f(t,x) = 1$ ), otherwise it must be that  $F(t,x,n) = 0$  meaning that there is no path.

4. TPBus – a function that provides Token paths from a given Transition to several Places and is denoted by  $G$ . Hence:

$$G: A_{Ti} \times A_T \times A_P \rightarrow A_V$$

where  $G(t,n,x) = 1$  means that there is a bus path from Transition  $n$  to Place  $x$  at time  $t$ . A constraining equation will allow this to occur only when both  $g(t,n) = 1$  and  $f(t,x) = 1$ , otherwise it must be that  $G(t,n,x) = 0$  meaning that there is no path.

5. Transition Firing – a function that causes the Tokens to move from Place to Place and is denoted by  $H$ . Thus:

$$H: A_{Ti} \times A_P \times A_{To} \rightarrow A_V$$

where for every value of in  $A_{Ti}$  and for every  $p$  in  $A_P$  there is exactly one  $k$  in  $A_{To}$  such that  $H(t,p,k) = 1$ . This means that Place  $p$  has  $k$  Tokens. For every other  $j$  in  $A_{To}$ ,  $H(t,p,j) = 0$ . We note that only enabled Transitions may fire or execute.

For instance, Figure 2 below illustrates a Dynamic Petri Net environment. A circle represents a Place and the box represents a Transition. The outline indicates that the Place or Transition is alive; its absence indicates that the Place or Transition is dead. Here, in Figure 2a, Places  $x$  and  $y$  as well as Transition  $n$  are not alive, but are dead. Diagrams, b, c, d illustrate the successive enabling of Place  $y$ , then Transition  $n$ , and finally Places  $x$ .

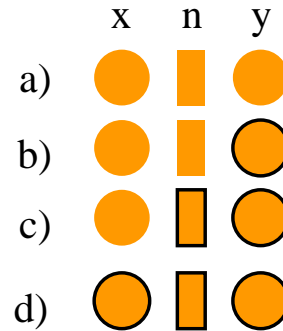


Figure 2. DPN with Alive and Dead Places/Transitions.

As another example, in Figure 3i, all Transitions and Places are enabled, indicated by dark lines around the squares and circles and the solid lines interconnecting the Places and Transitions. This diagram represents "The Universal Dynamic Petri Net", meaning that there are no other possible Places or Transitions allowed to be created. However, deletion of Places or Transitions is allowable. Dotted lines indicate removal of the arc connecting a Place or Transition.

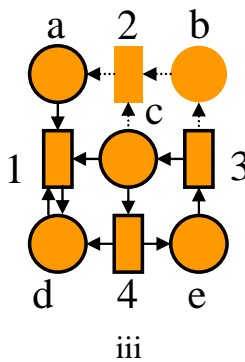
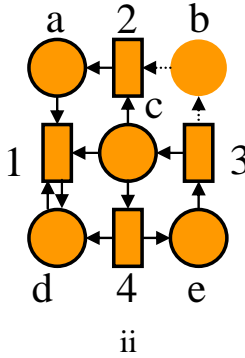
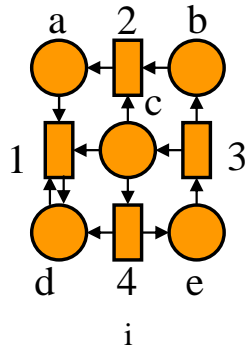


Figure 3. Dynamic Petri Net with (Recursive) Place / Transition Removal

In the example above, if Place b dies, as indicated by dotted lines around this Place, and illustrated in Figure 3ii then the TPBus from Transition 3 is removed as well as the PTBus connecting Transition 2. Moreover, if Transition 2 is then removed, the TPBus into Place a and

the PTBus from Place c is also removed. The final result is illustrated in Figure 3iii.

#### 4.5 DPN Marking and Reachability Tree

The Marking of a PN is typically defined as a fixed size vector, whose size is determined by the number of Places in the PN. The elements of the vector are typically positive integers. However, in our case, the Marking must represent the fact that Places may be added or deleted. Hence, the Marking vector length is fixed at  $|P_{max}|$ . We also need a way to represent Places that are not alive in the DPN. We use the symbol "-" in the Marking vector to indicate that that particular Place is inactive (or dead) and not involved with the DPN activities. As previously mentioned, the resident Tokens in the Places that are deleted may be either lost or retained for possible re-introduction to the DPN that the associated Place is made active again.

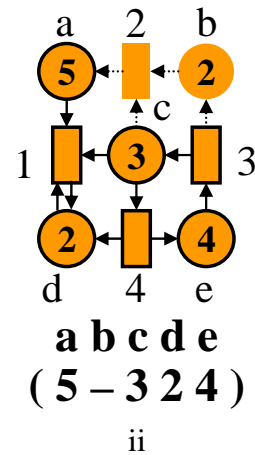
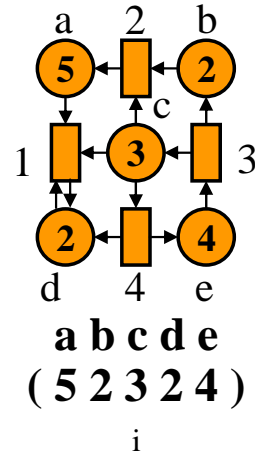


Figure 4. Dynamic Petri Net Marking Under Place Removal

In Figure 4i the same DPN is given as that illustrated in Figure 3i. However, now the Tokens have been assigned

to the Places. This is called the Marking of the DPN and is indicated by placing a non negative integer within the circle representing the respective Place. It is again illustrated as a Marking vector whose tuple provides the number of Tokens assigned to the respective Places. Indeed, the five tuple: (5 2 3 2 4) is shown in Figure 4i.

Figure 4ii illustrates the tuple assignment after Place b died. This assignment is used in subsequent operations involving the network. For instance, since all the Places in the above diagram have a non zero number of Tokens, any of the Transitions 1, 3, or 4 can fire. The usual method of illustrating the firing is given using a tree structure below in Figure 5. The DPN illustrated in the top of this diagram is precisely the same as the one shown in Figure 4ii.

Figure 5 illustrates the Reachability Tree and the "final" Token count per each Place. When Transition 1, or Transition 3 or Transition 4 fire, the Token count in each Place for each possible Transition execution is shown in Figure 5.

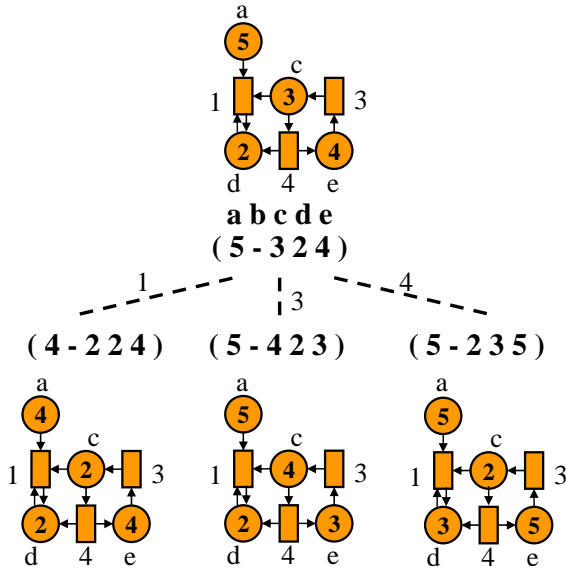


Figure 5 Reachability Tree for a DPN

## 5. APPLICATION OF DPN TO A TOPOLOGY REPRESENTATION

### 5.1 Background – Using the Directed Graph (DG) Model

It is well known that packet communication network topology is typically represented by directed graphs (DG). Under this scheme, the node of the graph represents the packet switch (or packet radio), while the arc (or edge) of the graph represents the communication channel between two nodes or radios. As packets move

through the network, they traverse from node to node through the connecting edges or arcs.

In order to provide a concrete example, we provide Figure 6, as the traditional directed graph model of the broadcast radio network. In that figure, we show 5 nodes, represented by the circles, and 6 edges, as represented by the directed arcs. Thus  $G(V, E)$  is the directed graph for the network, where  $V$  is the set of nodes, and  $E$  is the set of edges. We call this model as the DG model, or directed graph model. However, since each edge is associated with a single source node, and a single destination node, we cannot readily distinguish a point-to-point channel from a broadcast radio one.

In the figure, we desire a broadcast radio channel from node 1 to nodes 2, 3, and 4 as the set of edges  $e_1$ ,  $e_2$ , and  $e_3$ . Thus, a single packet from node 1 would have to be associated with a packet flowing simultaneously on the three edges  $e_1$ ,  $e_2$ , and  $e_3$ . Clearly this is not easily represented by the directed graph model.

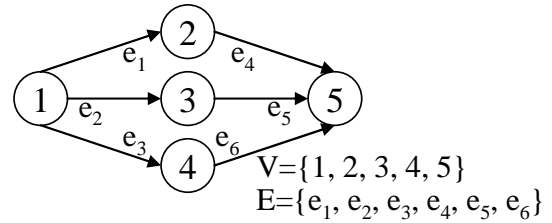


Figure 6. Directed graph model of a network

There is another difficulty with the simple graph theoretic representation for the broadcast radio environment, and that is the potential for interference. Specifically, when one radio transmits a packet, all other radios within its one hop communication range are capable of receiving it. If one of these one hop neighbors is also receiving a packet from a third transmitter, then interference will result, and neither packet may be correctly received. This interference which is a result of a node receiving two or more simultaneous transmissions is not easily represented with a simple directed graph model. In Figure 7, we assume that nodes 2, 3 and 4 have packets to be transmitted, and they collide at the receiving node, node 5. This is represented in Figure 7 pictorially, although we cannot represent this mathematically.

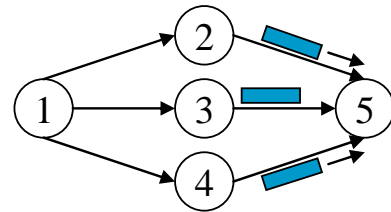


Figure 7. Packet collision at node 5

## 5.2 The DPN Model

In Figure 8, we show the equivalent DPN model of the network shown in Figure 6. Here, the nodes in the DG model are replaced one-to-one (i.e. in one-to-one correspondence) by the Places in the DPN. The broadcast radio channel is modeled as DPN transition, where the in-arc is an arc from the source Place to the Transition, and a set of out-arcs from the Transition leads to each of the one-hop neighbors of the source Place. Thus, the DPN model is very similar to the DG model, with the major difference being the replacement of the arcs by a transition.

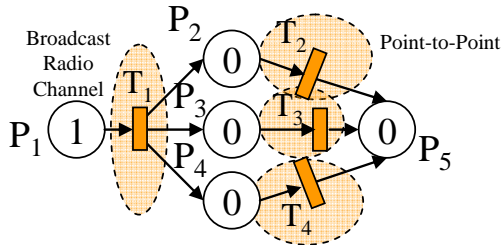


Figure 8. DPN model of a network.

The broadcast nature of the channel can readily be demonstrated as follows. We represent a packet as a Token that resides in a Place in the DPN. With a single Token in the Place  $P_1$ , the Transition  $T_1$  is enabled and capable of firing or being executed. When  $T_1$  does fire or execute, then the single token in Place  $P_1$  is removed, and simultaneously a token is deposited in each Place  $P_2$ ,  $P_3$ , and  $P_4$ . With the simultaneous creation of a Token in each of the receiving Places, we can accurately represent the simultaneous reception of an identical packet at

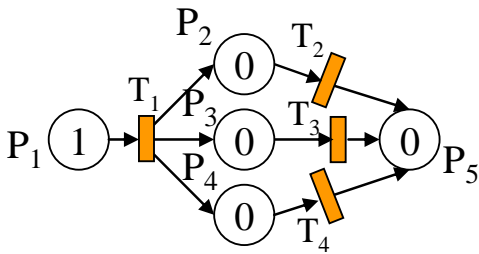


Figure 9. Ad hoc network with  $P_1$  ready to transmit.

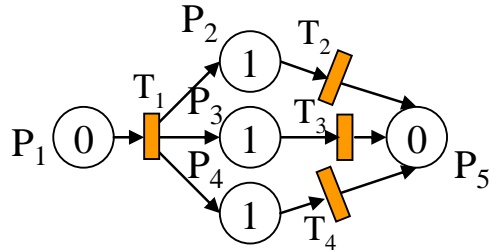


Figure 10. Ad hoc network after  $P_1$  transmits the packet.

each receiving radio node. This is shown at Figure 9,

prior to the packet transmission, while Figure 10 indicates the simultaneous packet reception at nodes (or Places)  $P_2$ ,  $P_3$ , and  $P_4$ . As before, the integers inside the Place represent the number of Tokens, or packets in that Place.

Using the DPN model, we can readily model effects of packet collisions or interference. In Figure 10, we assume that Places  $P_2$ ,  $P_3$ , and  $P_4$  each have a packet to be transmitted. Thus, Transition  $T_2$ ,  $T_3$ , and  $T_4$  are ALL enabled (capable of executing). If during a single timeslot or execution period they all did fire, then each of the tokens in  $P_2$ ,  $P_3$ , and  $P_4$  would be removed, and three Tokens would be therefore deposited in Place  $P_5$ . Since Place  $P_5$  initially had zero Tokens and contained three new tokens after the Transitions fired, we can readily see that a (multiple) packet collision has occurred, as shown in Figure 11.

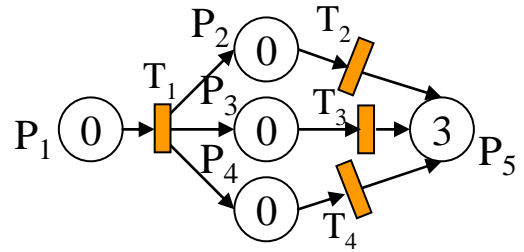


Figure 11. Ad hoc network showing packet collision at node (or Place)  $P_5$ .

We can also use the DPN to represent the finite error rate of packet channels, by associating with each arc exiting a Transition, a non-negative value that could be used to represent the probability of a packet loss or bit error. Space limitations prevent us from further presentation of this capability.

In addition, we can also use the DPN to represent node mobility, or node destruction. To represent a node being lost, destroyed, or inoperative, we simply remove that Place from the DPN, as well as any entering or exiting arcs. If the resulting configuration of the DPN contains any Transitions that have only inputs or outputs or neither, then it should also be removed from the DPN configuration. The loss of a communications link is similarly represented by removal of the associated arc.

## CONCLUSIONS/BENEFIT TO THE WARFIGHTER

We have presented an extension to Petri-Nets that can be used to model dynamic system behavior by structural changes in the PN. A mathematical formalization was presented that highlights this new behavior in both traditional as well as Many-Sorted Algebra form. Finally, a simple model that represents the broadcast nature of the

radio channel was also presented.

It is well known that the Army's Future Combat System (FCS) relies heavily on ad hoc radio network communication. The precise and accurate mathematical representation of the communication network environment is critical to the proper and reliable design of both the communication network protocols as well as the network performance analysis as indicated by the recent research in (Network Council 2005). In that report, the modeling, simulation, testing, and prototyping of large networks is identified as a critical research area. It is hoped that this earlier research can be continued and contribute to the body of knowledge that can be used by the Army in the design, analysis and testing of large ad hoc packet networks that are and continue to be critical to the successful conduct of military operations.

### **ACKNOWLEDGEMENTS**

The author wishes to express his appreciation to Dr. Charles Giardina, of Stevens Institute of Technology and BAE Systems for his kind review and constructive comments on this paper. The author also wishes to thank Mr. David Yee for his comments and proofreading of this paper.

### **REFERENCES:**

1. Giardina, C., 2000; Many-Sorted Algebra for Image Processing, Proceedings of MILCOM 2000.
2. Murata, T., 1989; Petri-Nets: Properties, Analysis and Applications, Proceedings of IEEE Vol. 77, No. 4, April 1989, pages 541-580.
3. National Research Council, 2005; "Network Science", Committee on Network Science for Future Army Applications, Board on Army Science and Technology, Division on Engineering and Physical Sciences, The National Academies Press, 2005.
4. Sreenivas, R.S. 2006; "On Minimal Representations of Petri-Net Languages", IEEE Transactions on Automatic Control, Vol. 51, No. 5, May 2006, pages 799-804.
5. Zuraski, R., and Zhou, M. 1994; Petri-Nets and Industrial Applications", A Tutorial, IEEE Transactions on Industrial Electronics, Vol. 41, No. 6, December 1994, pages 567-583.